

K6300

user manual



Documentation Author

Pierre Bureau for K-Team S.A.
Ch. de Vuasset, CP 111
1028 Préverenges
Switzerland

email: info@k-team.com

Url: www.k-team.com

TRADEMARK ACKNOWLEDGMENTS:

IBM PC: International Business Machine Corp.

Macintosh: Apple Corp.

SUN Sparc-Station: SUN Microsystems Corp.

LabView: National Instruments Corp.

MatLab: MathWorks Corp.

Webots: Cyberbotics

Khepera: K-Team and LAMI

LEGAL NOTICE:

- The content of this manual is subject to change without notice.
- All effort have been made to ensure the accuracy of the content of this manual. However, should any error be detected, please inform K-Team S.A.
- The above notwithstanding K-Team can assume no responsibility for any error in this manual.

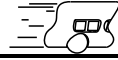
TABLE OF CONTENTS



1	Introduction	4
1.1	Safety Precautions	4
1.2	Unpacking and inspection	4
2	The K6300 Vision Turret	5
2.1	Overview	5
2.2	Running Modes	5
2.3	Optical System	7
3	Connections	8
4	Serial Communication Mode	10
4.1	Direct Communication	10
4.1.1	Testing the serial link	11
4.2	Communication through a Khepera	12
4.2.1	Detection Test	12
4.2.2	KNet protocol test	12
4.3	Stand Alone Mode	13
5	Programming	14
5.1	Control the Turret from a Khepera Program	14
5.2	Custom User Application and VVL Module System Calls	15
5.2.1	vvl_reset (bios call 368)	15
5.2.2	vvl_get_line_ptr (bios call 369)	15
5.2.3	vvl_get_mean_ptr (bios call 375)	15
5.2.4	vvl_get_mean_values (bios call 376)	15
5.2.5	vvl_get_image_ptr (bios call 370)	16
5.2.6	vvl_get_minmax (bios call 371)	16
5.2.7	vvl_get_min_prof (bios call 372)	16
5.2.8	vvl_get_max_prof (bios call 373)	17
5.2.9	vvl_aqu_image (bios call 374)	17
5.2.10	vvl_get_flag_ptr (bios call 377)	17
5.3	Communication with a Khepera Program	17
5.3.1	Using System Flags	18
5.3.2	Custom KNet commands	18
5.4	S Loader Mode	18
5.4.1	Serial link configuration	18
5.4.2	Starting the S loader	19
5.4.3	Loading an application file	19

6	Assembling Instructions	21
7	Serial Communication Protocol	22
8	KNet Communication Protocol	24

1 INTRODUCTION



1.1 Safety Precautions

Don't plug or unplug any connector or turret when the robot is powered (either with batteries or external power supply).

To prevent damage to the hardware, all cables and turrets must be properly plugged before switching the robot On, or before connecting power supply to the interface.

Switch Off the robot if not used during a day or longer.

Please unplug the power supply from the wall socket as well.

Avoid grabbing or touching the K6300 optical prism.

Do not try to remove the prism from its socket. Try to grab the turret using the electronic boards only.

1.2 Unpacking and inspection

Please check your package that should contain a K6300 vision turret and the documentation you are reading currently. Check the turret against figure 2.1, and especially the optical prism and lens. If you notice any damage, please contact your K-Team products dealer.



2.1 Overview

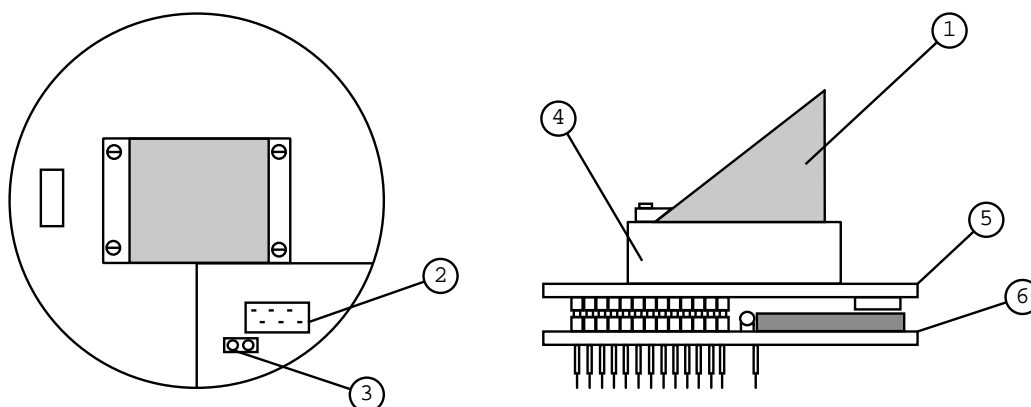


Figure 2.1: Overview of the K6300 vision turret.

Figure 2.1 is a general view of the turret when fully assembled, the followings details are pointed:

1. Optical prism.
2. Serial connector.
3. Jumper for stand alone mode.
4. Lens case.
5. K6300 vision board.
6. K6300 processor board.

The following sections detail the operations required to set up and tune the K6300 vision turret. Please refer to appendix 6 for instructions on turret assembling and disassembling.

2.2 Running Modes

The K6300 vision turret is equipped with a *Motorola* MC68331 processor, a RAM memory bank and a Flash memory bank. A completely independent Operating System is running on this CPU, it provides a complete set

of functions and a Khepera standard API to use and program the embedded vision system. Depending on the requested operating mode, different running modes are available.

To set up the turret running mode, the processor board should be first disassembled from the vision board (Refer to appendix 6 for instructions). The processor board is such as displayed on figure 2.2 and the following details are pointed:

1. Jumper for stand alone mode.
2. Serial connector.
3. Flash memory.
4. MC68331 processor.
5. Reset button.
6. Running mode jumpers.

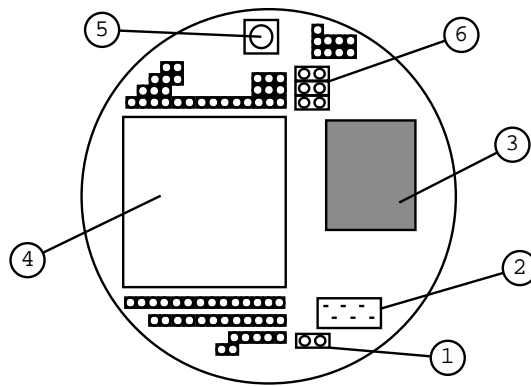


Figure 2.2: Overview of the MC68331 processor board.

The running mode is set by the jumpers position when resetting or switching the turret on. This will determine the serial communication speed and the operating mode for the turret.

Various jumper settings are detailed on figure 2.3 and refer to the following mode descriptions:

0. Unused.
1. Serial communication mode (9600 bps): Mode to control the turret using the Serial communication protocol.
2. Serial communication mode (19200 bps): Same as mode 1.
3. Serial communication mode (38400 bps): Same as mode 1.
4. User Application mode (9600 bps): Start an application stored in the turret's non volatile memory. The application should be flashed first, using the S loader (see section 5.4 for details).

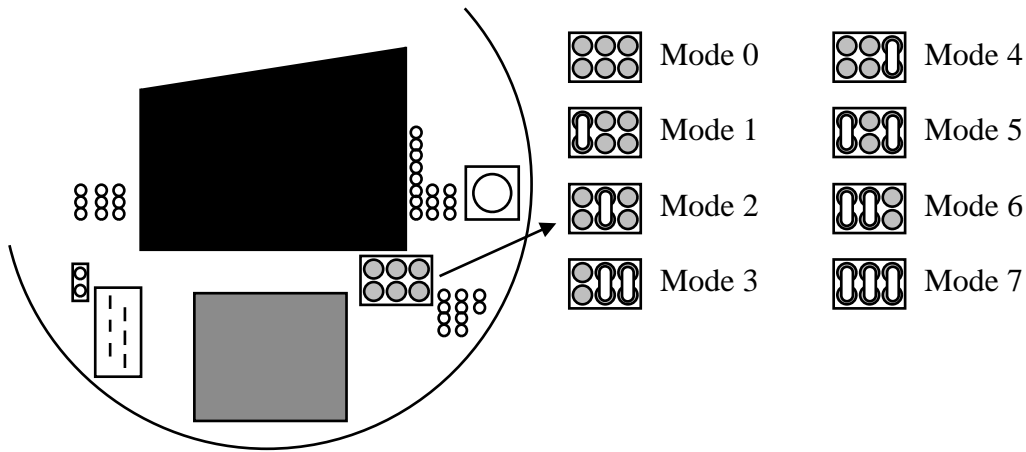


Figure 2.3: Jumper positions to select running mode

5. S Loader mode (9600 bps): The turret waits for an application to be transferred in RAM and executes it when fully uploaded.
6. S Loader mode (38400 bps): Same as mode 5.
7. Test mode (9600 bps): Successive tests are performed and the results are displayed using the serial link.

Please refer to chapter 3 for a detailed description of the Serial Communication mode and methods to use it with a Khepera robot. Refer to chapter 5 for instructions concerning turret programs and S Loader mode.

2.3 Optical System

The K6300 vision turret is based on an optical prism and a pin hole lens to focus the image on the vision sensor. The lens is adjusted for a correct image focus with objects placed 10-60cm away from the robot. In case of problem concerning the turret optical system, please contact your local K-Team product dealer.

3 CONNECTIONS



The K6300 vision turret is based on a MC68331 processor which is equipped with its own UART interface and thus, its own serial line. The Serial connector placed on the turret is **not** an access to the Khepera serial line as it is usually for other turrets. This serial connector is dedicated to serial communication with the turret itself.

This configuration enables different communication methods with the turret and different connections. There is a specific setup to communicate with the Khepera, to communicate with the turret and to use the turret as a stand alone system.

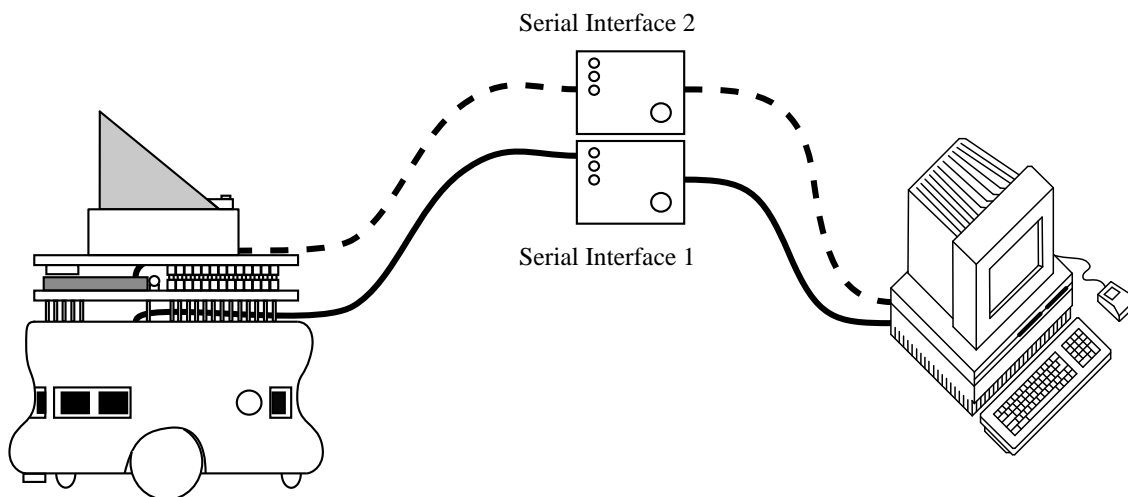


Figure 3.1: Serial connections to the turret and/or Khepera

The power supply jack can be connected to one interface or the other but **not** to both at the same time. If the Khepera is running autonomously, embedded batteries will supply the K6300 turret. In any case, one single power source will supply the entire system.

The first serial connection is linking the Khepera itself to the host computer. One of the host computer's serial port should be connected to the robot, using an interface-charger module. **Do not** use the turret serial connector. This connection can be required to send commands to the Khepera and/or to debug an user application running on the robot.

The second serial connection is linking the K6300 turret to the host computer. One of the host computer's serial port should be connected to the turret, using an interface-charger module. User applications can be

developped to run on the turret's processor. This connection can be used to debug these applications or to send direct commands to the turret for stand alone testing.

The K6300 vision turret is basicaly designed as an autonomous vision system for the Khepera robot, therefore, common setup should not require a serial link. The turret and the robot can communicate through the KNet bus, exchanging informations to build up an effective dual processor system.

4 SERIAL COMMUNICATION MODE



Even though the K6300 has been designed to be used as an autonomous embedded vision system, serial communication with the turret and/or the Khepera can be usefull for development and debbuging purpose. The serial line setup and terminal configuration should be the same as the one used for Khepera serial communication (please refer to the Khepera User Manual for details).

Usual commands can be sent to the Khepera robot and completly remote operation is still possible. Commands can be sent to the turret using the KNet protocol and the 'T' command as any common turret. Moreover, some commands can be directly sent to the turret using the turret's serial port.

4.1 Direct Communication

To enable direct communication between the K6300 vision and the host computer, a serial line must be connected from the turret serial connector (see section 2.1) to the computer, using a k-team serial interface. This serial interface is provided with a Khepera robot package.

A terminal emulator should be running on the host computer, make sure the cable is connected to the correct serial port. Terminal configuration should be set to **8 bit data, 1 start bit, 2 stop bit and no parity**.

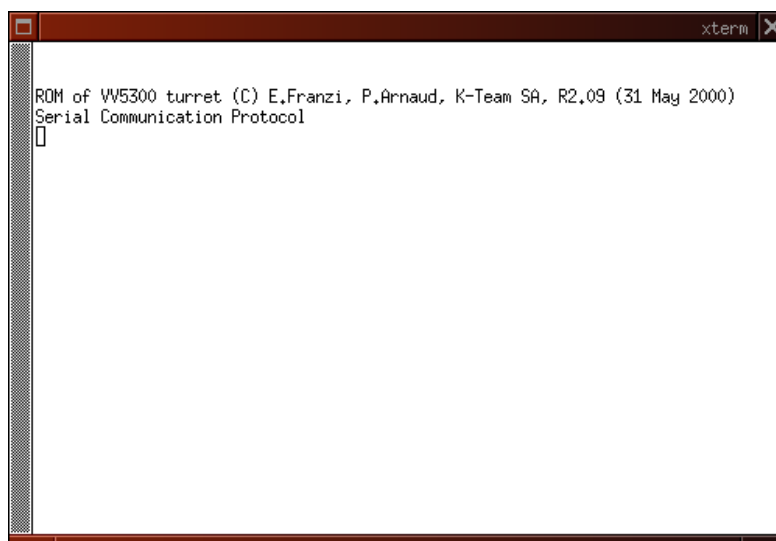


Figure 4.1: K6300 boot message

The turret's running mode sets communication speed. **Default mode for a turret is Serial communication 38400 bps.** Any Serial communication mode can be used according to the desired communication speed (see section 2.2 for details).

When switching the system power on, the turret boot message should be displayed on the terminal as on figure 4.1. If the message is not displayed, check the serial connection and the turret running mode. If random characters are displayed, check the serial communication speed according to the turret running mode.

4.1.1 Testing the serial link

If the boot message is correctly displayed, the turret is ready to receive serial commands. A complete list of commands is available in appendix 7. The 'B' command can be sent to test the turret connection. If the answer is correct, an image acquisition test should be performed. The 'Q' command should be sent to acquire a new image, and the 'X' command should start the download. A long serie of characters should be displayed on the terminal, these are the image raw data received from the turret.

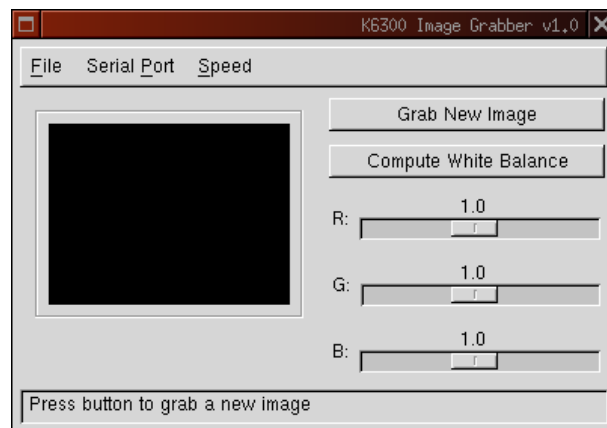


Figure 4.2: K6300 image grabber

The image data need a little bit of processing to be displayed. Please refer to the vv6300 CMOS camera datasheet (available from www.k-team.com), for a detailed description of the image format. A test application is also available for download from *K-Team* website to display K6300 images (see figure 4.2).

4.2 Communication through a Khepera

Commands can be sent to the turret using the KNet protocol as any usual turret. The serial link between the Khepera and the host computer must be set up as usual depending on the robot's running mode (see the Khepera User Manual for details). The serial cable can be connected to the Khepera itself or to any turret **except** the K6300 vision turret.

When switching system power on, the Khepera boot message should be displayed on the terminal. Normal Khepera commands can be sent and the robot should react as usual.

4.2.1 Detection Test

Once the serial link between the Khepera and the host computer is valid, the "net" command should be used to detect all turrets plugged on the KNet. The K6300 vision turret must be detected as displayed on figure 4.3.

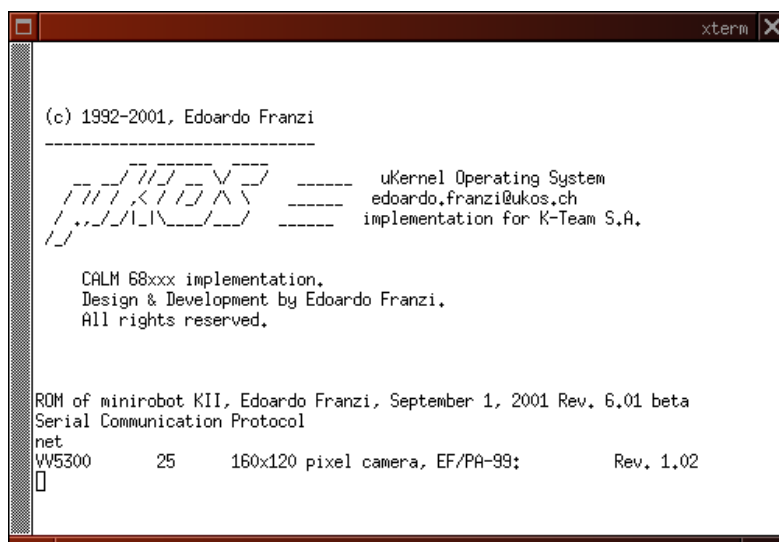
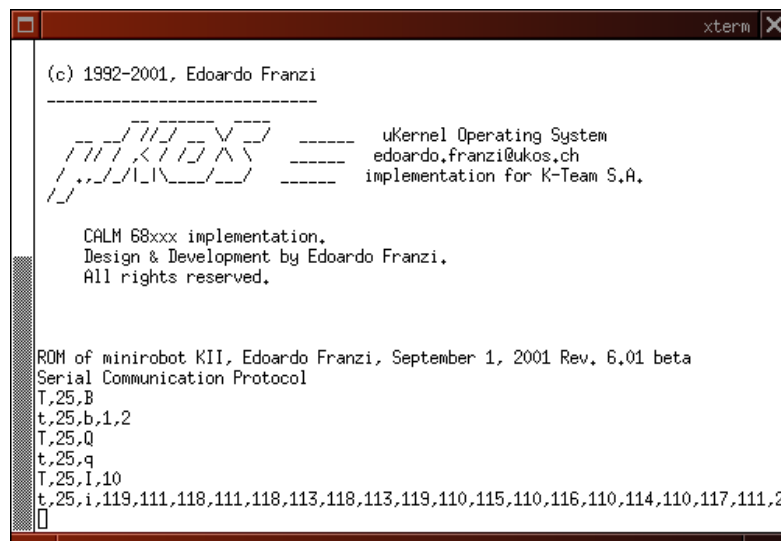


Figure 4.3: Detection test result

4.2.2 KNet protocol test

If the turret is correctly detected, serial commands can be tested. The usual ‘T’ syntax must be used to send commands through the Khepera to the turret. Anyone of the supported KNet commands can be executed and should return the corresponding answer, please refer to appendix 8 for further detail.

The most basic test is to use the ‘B’ command to get the turret software version. Typing ‘T,25,B’ should launch the command then both version and revision number should be returned. More complex interactions require parameters, for instance to read line 10 from the acquired image. Image acquisition is done after sending a ‘T,25,Q’ command. The required line is returned by typing ‘T,25,I,10’.



4.3 Stand Alone Mode

The serial communication is exactly the same as described in section 4.1. The turret does not need to be plugged on a robot, all the regular turret features can still be used. The only difference is a jumper setting required to provide power to the turret. A regular 2.54mm jumper must be plugged in the correct socket (ser item 3 on figure 2.1).



Two main control modes are available to use the K6300 vision turret. First, it can be controlled as a regular Khepera turret. In this case, a program running on the robot can interact with the turret using KNet commands. Second, it can be programmed to match application specific needs, and communicate with the robot using a customized protocol.

5.1 Control the Turret from a Khepera Program

The communication between a Khepera program and the K6300 turret respect basically the rules described in section 4.2. The program can call any supported KNet commands (see appendix 8) by sending the correct sequence to the turret. The corresponding answers, received from the KBus, can be processed as required by the application.

The following piece of code is one method to send a command to the turret:

```
/* K-Bus message syntax:
 * byte1 : turret ID
 * byte2 : size of the following in bytes
 * byte3 : the command
 * byte4 : first command argument
 * ...   :
 * byteN : command argument N
 */

uint8 message[] = {25,1,'Q'};
uint32 size = 2;
int32 status;
uint8 ans[1];

[...]

while(msg_reserve_channel(0))
    tim_switch_fast();

/* Send a 'Q' command */
status=msg_send_message(message,size);

/* Read the acknowledge */
status=msg_receive_message(ans,1);
if(!status)
    printf("Recv: %c\n\r",ans[0]);

msg_release_channel(0);
```

5.2 Custom User Application and VVL Module System Calls

The K6300 turret is able to download and execute a custom user application, compiled with the appropriate software package. Such an application can be written in C or assembly language, and it can use the system BIOS library.

The K6300 BIOS provides the TIM, VAR, STR, SER and COM modules as well as the specific vision VVL module. A detailed description of the BIOS API can be found in the *Khepera BIOS Reference Manual* and the VVL module is described in the following section.

A turret application structure is the same as a standard Khepera application, it should be using tasks and similar BIOS utilities. It should be compiled using the appropriate software package, either using KTPProject or a correct Makefile. While a program is running on the turret, **it is still receiving and processing serial commands from the KNet.**

5.2.1 vvl_reset (bios call 368)

```
int32 vvl_reset (void);
```

Call this function once before using any other vvl... function. This initialises the module and sets up the hardware interface and the camera.

5.2.2 vvl_get_line_ptr (bios call 369)

```
const uint8* vvl_get_line_ptr (int32 line);
```

This function returns a pointer to the specified line of 160 pixels. The line argument must be in the range 0..119. If outside of this range, the function returns a pointer of value 0xFFFFFFFF (or -1).

5.2.3 vvl_get_mean_ptr (bios call 375)

```
const uint8* vvl_get_mean_ptr (void);
```

This function returns a pointer to 120 pairs of mean pixel values. The pair consists of the mean value of left half (pixels 0 through 79) and the mean value of the right half (pixels 80 through 159) of each line.

To access the left mean value of line number 25, you would access byte number $2*25+0$. To access the right mean value of the same line, you would access byte number $2*25+1$.

5.2.4 vvl_get_mean_values (bios call 376)

```
uint16 vvl_get_mean_values (void);
```


This function returns the overall mean values of the left half and of the right half of the image. Each half is 80 pixels wide and 120 pixels high. The left and right values are stored as the high and low byte of the returned value. To extract them, use the following piece of code :

```
uint16 mean_pair = vvl_get_mean_values ();
uint8 left_mean = (uint8)(mean_pair >> 8);
uint8 right_mean = (uint8)(mean_pair);
```

5.2.5 vvl_get_image_ptr (bios call 370)

```
const uint8* vvl_get_image_ptr (void);
```

This function returns a pointer to the raw array of 160 x 120 bytes, directly encoding the individual pixel values. To access the pixel located at the position [x;y], use the following piece of code :

```
const uint8* ptr = vvl_get_image_ptr ();
...
uint8 pixel = ptr[x+y*160];
```

5.2.6 vvl_get_minmax (bios call 371)

```
uint32 vvl_get_minmax (void);
```

This function returns the co-ordinates of the minimum intensity pixel and of the maximum intensity pixel. The co-ordinates are encoded in the 32-bit result as [xmin;ymin;xmax;ymax]. Use the following piece of code to extract the individual co-ordinates :

```
uint32 encoded = vvl_get_minmax ();
uint8 x_min = (uint8)(encoded >> 24);
uint8 y_min = (uint8)(encoded >> 16);
uint8 x_max = (uint8)(encoded >> 8);
uint8 y_max = (uint8)(encoded);
```

5.2.7 vvl_get_min_prof (bios call 372)

```
const uint8* vvl_get_min_prof (uint8 max_level);
```

This function computes the minimum profile of each column (there is a total of 160 columns). It searches the pixel with the lowest intensity and stores its position in the column (in the range 0..119). If the intensity of the given pixel is above the maximum level, then its position is discarded and 255 is stored instead. If every pixel should be considered, set max_level to 255.

The following piece of code prints the [y] position of each minimum pixel value, column by column :

```

uint32 x;
const uint8* profile = vvl_get_min_prof (255);
for (x = 0; x < 160; x++) {
    printf ("x=%3d => y=%3d\n", x, profile[x]);
}

```

5.2.8 vvl_get_max_prof (bios call 373)

```
const uint8* vvl_get_max_prof (uint8 min_level);
```

This function computes the maximum profile of each column (there is a total of 160 columns). It searches the pixel with the highest intensity and stores its position in the column (in the range 0..119). If the intensity of the given pixel is below the minimum level, then its position is discarded and 255 is stored instead. If every pixel should be considered, set min_level to 0. See vvl_get_max_prof for an example.

5.2.9 vvl_aqu_image (bios call 374)

```
void vvl_aqu_image (void);
```

This function takes a snapshot with the camera. The acquisition of the image takes about xx us. You must call this function before you call one of the following functions, or their result will be undefined :

- vvl_get_line_ptr
- vvl_get_mean_ptr
- vvl_get_mean_values
- vvl_get_image_ptr
- vvl_get_min_prof
- vvl_get_max_prof

5.2.10 vvl_get_flag_ptr (bios call 377)

```
uint8* vvl_get_flag_ptr (void);
```

This function returns a pointer to 64 uint8 flags (8-bit values) which can be freely used by the user. The host can check or modify these flags through the K-bus command F.

5.3 Communication with a Khepera Program

While an user application is running on the K6300 turret, it might need to communicate with the Khepera robot. All communication between the robot and the turret are based on the KBus protocol, an enhanced SPI bus.

The Khepera robot is always the bus master, and any turret is a slave. This is why any communication must be initiated from the robot.

A method to send KNet commands to the turret from a terminal is described in section 4.2. From a Program running on a Khepera, a simple communication method is to use the system flags. If the flags are not enough, KNet commands can be customized depending on the application requirements.

5.3.1 Using System Flags

A set of 64 system flag can be read and changed from both a Khepera program and a turret user application. These flag can be used, with other standard KNet commands, to set up a complex communication between the robot and the turret.

The robot can read and change the flags using the 'F' command. This command is send as any other KNet command. The turret application can read and change the flags using the `vv1_get_flag_ptr()` function.

5.3.2 Custom KNet commands

Customizing KNet commands is the next available solution to communicate with a Khepera program. The low level command table can be redefined so that whenever a KNet command is received, an user defined routine will be called instead of the standard system call.

As some low level definitions are necessary to redefine the command table, the K6300 program template should be used as a starting base for any application requiring this feature. This template can be downloaded from K-Team website and further information is available within the Template itself.

5.4 S Loader Mode

After being succesfully compiled, an user application must be uploaded in the turret memory to be executed. Two methods are available to start the Sloader, and upload the application.

5.4.1 Serial link configuration

The serial link has to be properly configured to enable a turret computer communication. Please refer to section 3 for a detailed description of necessary connections and configuration. When booting in S loader mode, the serial communication speed is set at boot time, and the host terminal speed must be set according to the chosen mode. Other parameters are set to the usual value.

5.4.2 Starting the S loader

Two different methods are available to start the S loader. First, when booting the turret in one of the S loader modes (5 or 6), the loader is automatically started. The communication speed is set according to the chosen mode, and the entire loading process will use this baudrate. The host terminal terminal display should look as figure 5.1.

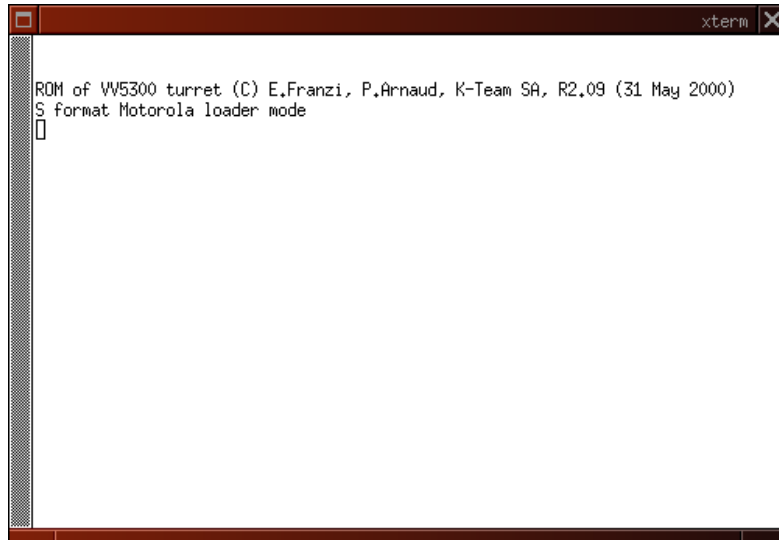


Figure 5.1: Sloader mode at boot time

Second, the S loader can be started using the serial communication protocol. The command "run sloader" is used to start the loader, and the transfer speed will stay the same as the serial communication speed. The host terminal terminal display should look as figure 5.2.

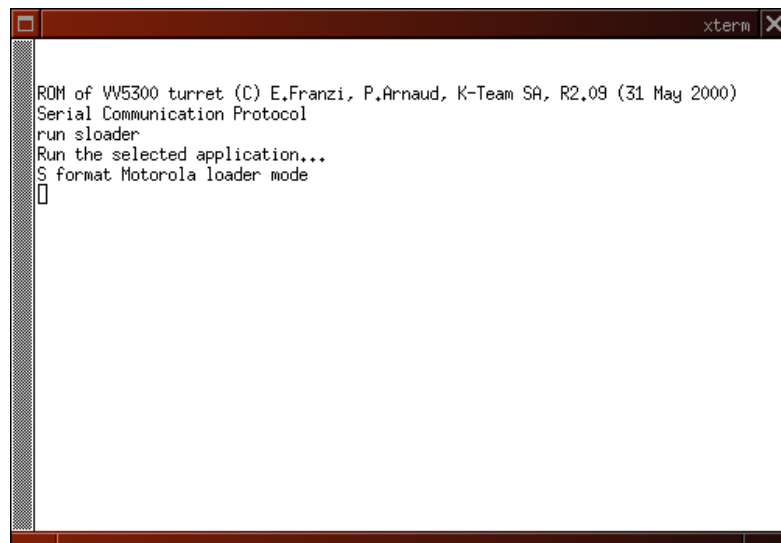
5.4.3 Loading an application file

Once started, the S loader simply waits for an executable file to be transferred through the serial line. Depending on your system and on the terminal emulator you are using, several methods are available to send a file. The most common is to use a "send file" command from the terminal emulator.

As soon as the loading process is initiated, one of the turret's led indicator is switched on. The indicator should stay on during the entire loading process and turned off when the download is completed. The downloaded application is executed as soon as the transfer is achieved and the following message is displayed:

S: download terminated

In case of problem loading an application, check all the connections and

A screenshot of an xterm window with a dark red title bar and a grey border. The window contains the following text:

```
ROM of VV5300 turret (C) E.Franzi, P.Arnaud, K-Team SA, R2.09 (31 May 2000)
Serial Communication Protocol
run sloder
Run the selected application,..
S format Motorola loader mode
█
```

Figure 5.2: Sloder mode from serial communication

configuration and try to use the serial communication protocol using the same baudrate.

6 ASSEMBLING INSTRUCTIONS



Assembling and disassembling the turret are delicate operations. Please operate very carefully to avoid breaking or bending the connection pins.

Assembling is the easiest part of the job. Simply make sure the pins are correctly aligned with the corresponding socket and press gently the top board.

Disassembling requires a plastic tool to use as a lever. It is easy to bend pins by trying to remove the top board too quickly. The best method to operate safely is to slightly unplug the board on one side using the lever, then slightly unplug the opposite side. After a few press on each side, the boards should be loose enough for an easy removal.

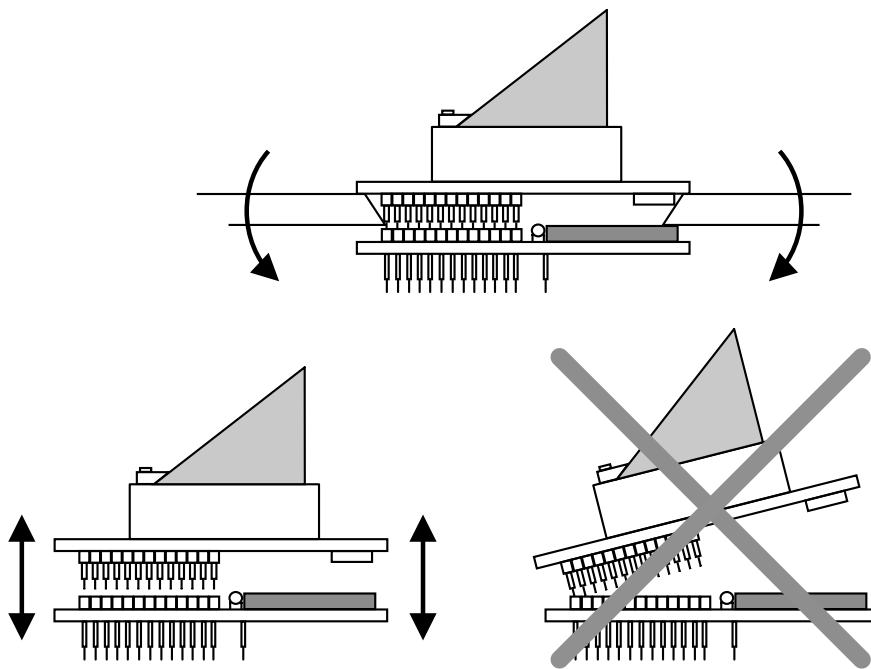


Figure 6.1: Assembling and disassembling the turret

7 SERIAL COMMUNICATION PROTOCOL

B Read software version

Command: B
Answer: b, bios_version, protocol_version
Effect: Read software version stored in the turret's non-volatile memory.

I Read image line

Command: I, line
Answer: i, pixel0, pixel1, pixel2, ... pixel158, pixel159
Effect: Return the 160 pixels of the given line. A decimal value is returned for each pixel of the line, separated with commas. An image must be acquired first.

L Set LED state

Command: L, led, state
Answer: l
Effect: Set the given LED state. Each LED can be switched on (1) or switched off (0).

M Get min and max pixels

Command: M
Answer: m, min_x, min_y, max_x, max_y
Effect: Return coordinates, for minimum intensity pixel and maximum intensity pixel. An image must be acquired first.

N Read one fourth of image line

Command: N, line
Answer: n, pixel0, pixel1, pixel2, ... pixel38, pixel39
Effect: Return one pixel out of four for the given line. A decimal value is returned for each of the 40 pixels, separated with commas. An image must be acquired first.

O Get image minimum profile

Command: O, threshold
Answer: o, column0, column1, column2, ... column158, column159
Effect: Return the computed minimum profile for each column. see section 5.2.7 for details. An image must be acquired first.

P Get image maximum profile

Command: P, threshold
Answer: p, column0, column1, column2, ... column158, column159
Effect: Return the computed maximum profile for each column. see section 5.2.8 for details. An image must be acquired first.

Q Acquire new image

Command: Q
Answer: q
Effect: A new image is acquired.

S Average value for half a line

Command: S, line
Answer: s, mean_first_half, mean_second_half
Effect: Compute the average value for each half of the given line.

X Transfer image data

Command: X
Answer: special
Effect: Send the complete binary image data to the serial line. An image must be acquired first.

8 KNET COMMUNICATION PROTOCOL



B Read software version

Command: B
Answer: b, bios_version, protocol_version
Effect: Read software version stored in the turret's non-volatile memory.

F Read and Change system flags

Command: F, flag+operation_code
Answer: f, flag_value
Effect: Read a flag and possibly modify it. This command has a single argument, consisting of the flag number added with an operation code. The flag number is in the range 0..63 and the operation code can be either +0 (just read the flag), +64 (read and set the flag), +128 (read and clear the flag), +192 (read and change the flag). (ie 'F,72' will read and set flag number 8).

I Read image line

Command: I, line
Answer: i, pixel0, pixel1, pixel2, ... pixel158, pixel159
Effect: Return the 160 pixels of the given line. A decimal value is returned for each pixel of the line, separated with commas. An image must be acquired first.

O Get image minimum profile

Command: O, threshold
Answer: o, column0, column1, column2, ... column158, column159
Effect: Return the computed minimum profile for each column. see section 5.2.7 for details. An image must be acquired first.

P Get image maximum profile

Command: P, threshold

Answer: p, column0, column1, column2, ... column158, column159

Effect: Return the computed maximum profile for each column. see section 5.2.8 for details. An image must be acquired first.

Q Acquire new image

Command: Q

Answer: q

Effect: A new image is acquired.



K-Team SA
1028 Préverenges
CH DE VUASSET, CP 111
SWITZERLAND
